A Game Theoretic Analysis of Selfish Mobile Computation Offloading

Slađana Jošilo, György Dán

Department of Network and Systems Engineering
School of Electrical Engineering, KTH Royal Institute of Technology

Atlanta, May 4, 2017

Mobile Edge Computing (MEC)

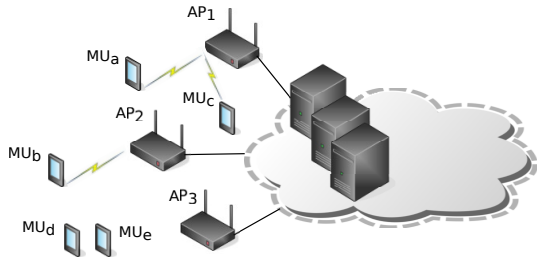
Mobile Edge Computing (MEC)

Enabler of 5G that provides

Mobile Edge Computing (MEC)

Enabler of 5G that provides

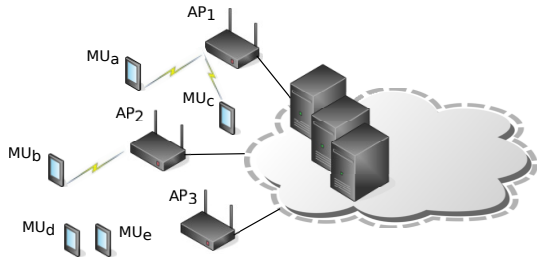
- computing resources and high bandwidth
- low latency
- real-time access to radio network information
- collaboration between mobile operators and application providers



Mobile Edge Computing (MEC)

Enabler of 5G that provides

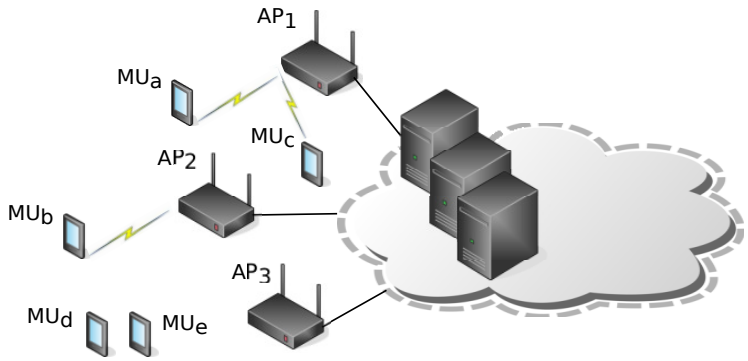
- computing resources and high bandwidth
- low latency
- real-time access to radio network information
- collaboration between mobile operators and application providers



Important question

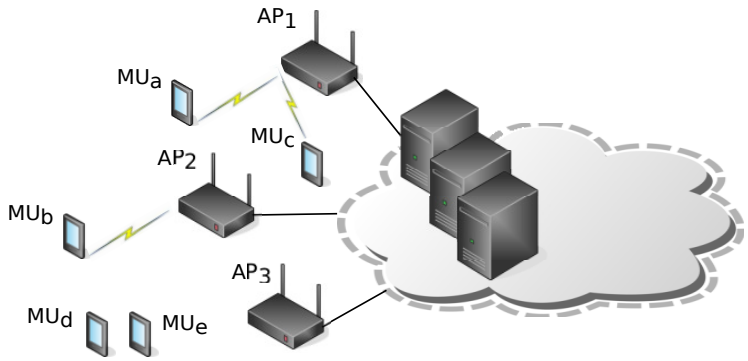
How to utilize mobile edge resources efficiently?

Mobile Cloud Computing System



- Cloud server
- Set of APs
  $\mathcal{A} = \{1, 2, ..., A\}$
- Set of MUs
  $\mathcal{N} = \{1, 2, ..., N\}$

Mobile Cloud Computing System

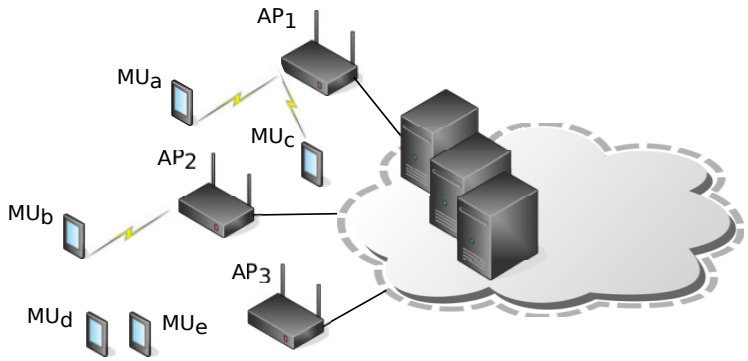


- Cloud server
- Set of APs
  $\mathcal{A} = \{1, 2, ..., A\}$
- Set of MUs
  $\mathcal{N} = \{1, 2, ..., N\}$

Computation Offloading

- Task of MU i, $< D_i, L_i >$
  - size of the input data D_i
  - computational complexity L_i

Mobile Cloud Computing System

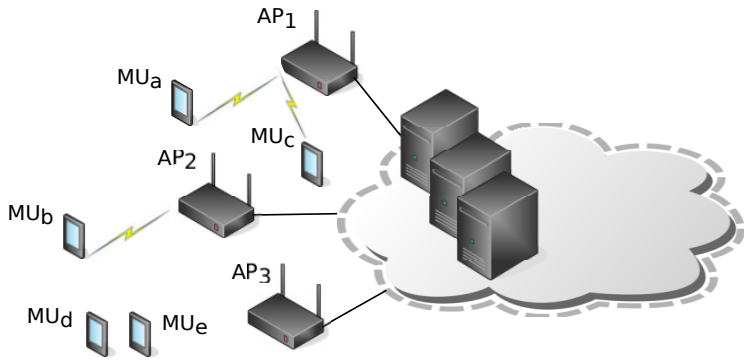


- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of MUs $\mathcal{N} = \{1, 2, ..., N\}$

Computation Offloading

- Task of MU i, $< D_i, L_i >$
  - size of the input data D_i
  - computational complexity L_i
- Decision of MU i is $d_i = \begin{cases} 0, & \text{if MU } i \text{ performs the task locally} \\ a, & \text{if MU } i \text{ offloads the task using AP } a \end{cases}$

Mobile Cloud Computing System

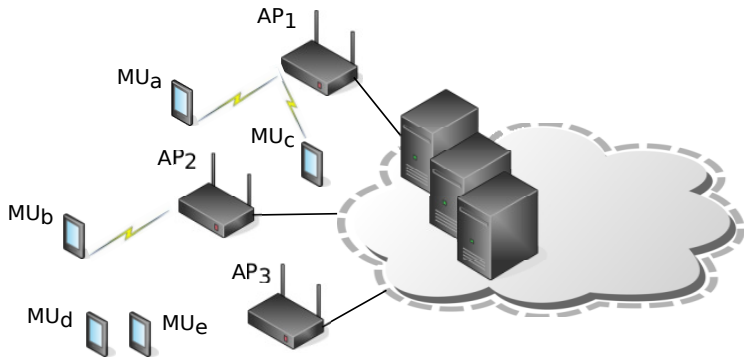


- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of MUs $\mathcal{N} = \{1, 2, ..., N\}$

Computation Offloading

- Task of MU i, $< D_i, L_i >$
  - size of the input data D_i
  - computational complexity L_i
- Decision of MU i is $d_i = \begin{cases} 0, & \text{if MU } i \text{ performs the task locally} \\ a, & \text{if MU } i \text{ offloads the task using AP } a \end{cases}$
- Set of decisions for all MUs is a *strategy profile* $\mathbf{d}$

Mobile Cloud Computing System

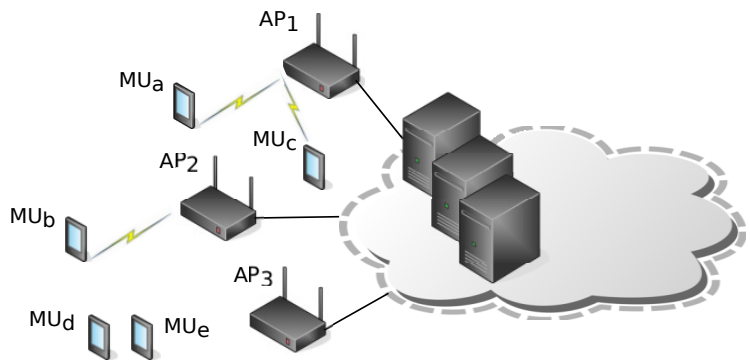


- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of MUs $\mathcal{N} = \{1, 2, ..., N\}$

Computation Offloading

- Task of MU i, $< D_i, L_i >$
  - size of the input data D_i
  - computational complexity L_i
- Decision of MU i is $d_i = \begin{cases} 0, & \text{if MU } i \text{ performs the task locally} \\ a, & \text{if MU } i \text{ offloads the task using AP } a \end{cases}$
- Set of decisions for all MUs is a *strategy profile* $\mathbf{d}$
- Number of MUs that offload via AP a in a strategy profile $\mathbf{d}$ is $n_a(\mathbf{d})$

Mobile Cloud Computing System

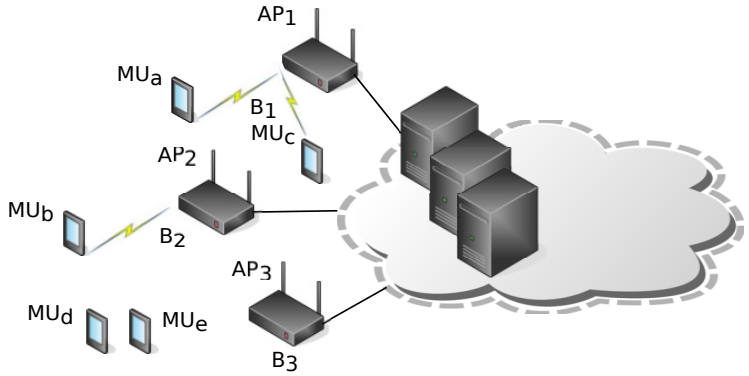


- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of MUs $\mathcal{N} = \{1, 2, ..., N\}$

Computation Offloading

- Task of MU i, $< D_i, L_i >$
  - size of the input data D_i
  - computational complexity L_i
- Decision of MU i is $d_i = \begin{cases} 0, & \text{if MU } i \text{ performs the task locally} \\ a, & \text{if MU } i \text{ offloads the task using AP } a \end{cases}$
- Set of decisions for all MUs is a *strategy profile* $\mathbf{d}$
- Number of MUs that offload via AP a in a strategy profile $\mathbf{d}$ is $n_a(\mathbf{d})$
- Total number of MUs that offload in a strategy profile $\mathbf{d}$ is $n(\mathbf{d})$

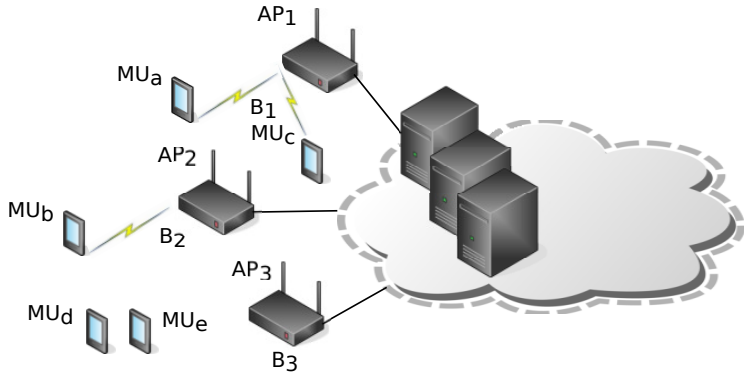
Communication Model



- Uplink rate of MU i via AP a

$$\omega_{i,a}(\mathbf{d}) = \frac{R_{i,a}}{n_a(\mathbf{d})}$$

Communication Model



- Uplink rate of MU i via AP a

$$\omega_{i,a}(\mathbf{d}) = \frac{R_{i,a}}{n_a(\mathbf{d})}$$

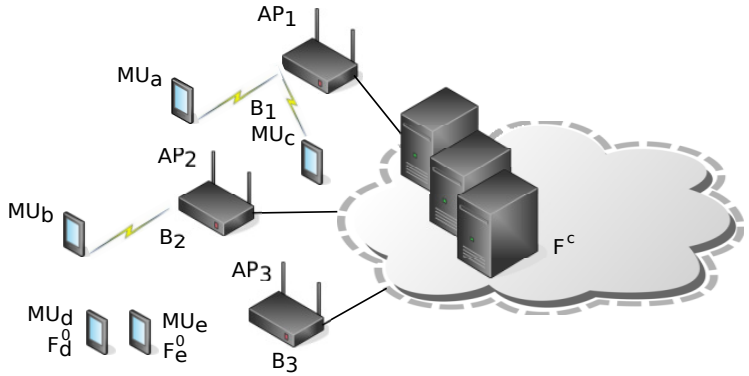
- Transmission time of MU i for offloading via AP a

$$T_{i,a}^{c,off}(\mathbf{d}) = \frac{D_i}{\omega_{i,a}(\mathbf{d})}$$

- Energy consumption of MU i for offloading via AP a

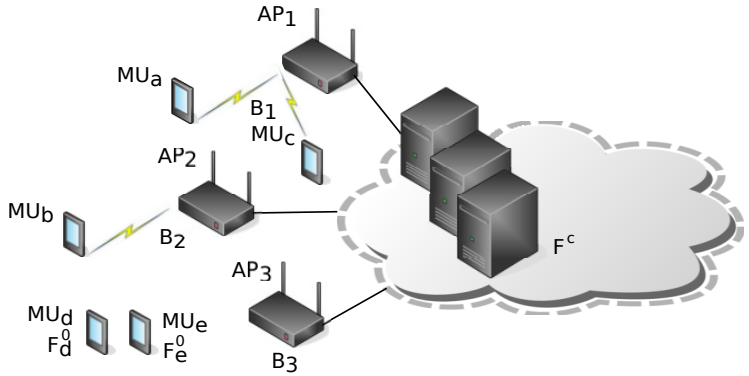
$$E_{i,a}^{c}(\mathbf{d}) = \frac{D_i P_i}{\omega_{i,a}(\mathbf{d})}$$

Computation Model



- Local Computing
  - F_i^0 is the computation capability of MU i
  - Execution time of the task
$$T_i^0 = \frac{L_i}{F_i^0}$$
  - Energy consumption
$$E_i^0 = v_i L_i$$

Computation Model

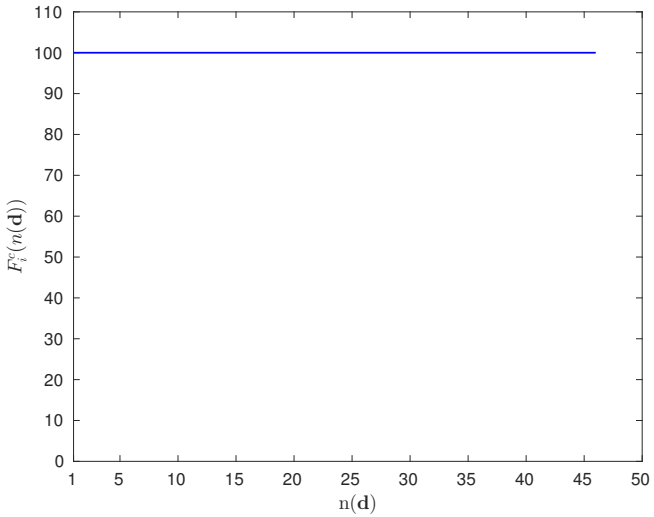


- Local Computing
  - F_i^0 is the computation capability of MU i
  - Execution time of the task
$$T_i^0 = \frac{L_i}{F_i^0}$$
  - Energy consumption
$$E_i^0 = v_i L_i$$

- Cloud Offloading
  - F^c is the computation capability of a cloud
  - MU i receives $F_i^c(n(\mathbf{d}))$ amount of computing power
  - Execution time of the task
$$T_i^{c,exe}(\mathbf{d}) = \frac{L_i}{F_i^c(n(\mathbf{d}))}$$

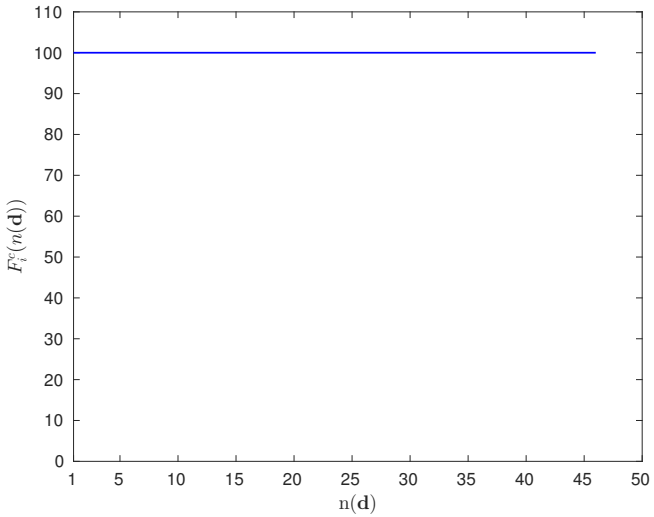
Cloud Model

Cloud Model

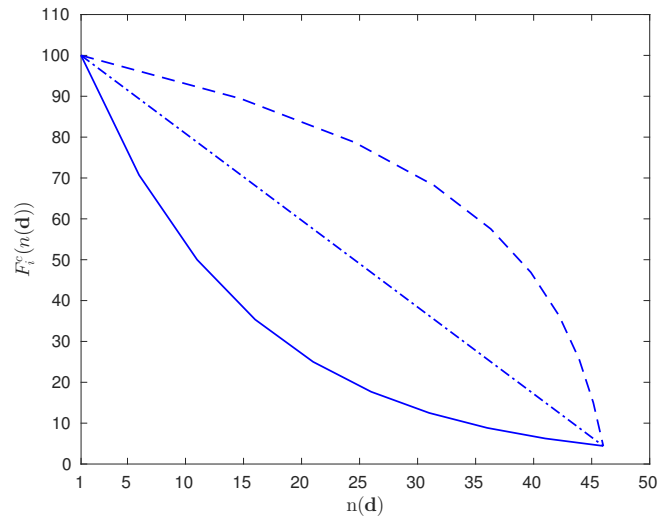


- Elastic cloud
  - $F_i^c(n(\mathbf{d}))$ - constant and equal to F^c

Cloud Model



- Elastic cloud
  - $F_i^c(n(\mathbf{d}))$ - constant and equal to F^c



- Non-elastic cloud
  - $F_i^c(n(\mathbf{d}))$ - decreasing function of $n(\mathbf{d})$

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T \underbrace{(T_{i,a}^{c,off}(\mathbf{d}) + T_i^{c,exe}(\mathbf{d}))}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,a}^c(\mathbf{d})}_{\text{energy}}$$

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T \underbrace{(T_{i,a}^{c,off}(\mathbf{d}) + T_i^{c,exe}(\mathbf{d}))}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,a}^c(\mathbf{d})}_{\text{energy}}$$

Joint computing and radio resource allocation problem

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T \underbrace{(T_{i,a}^{c,off}(\mathbf{d}) + T_i^{c,exe}(\mathbf{d}))}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,a}^c(\mathbf{d})}_{\text{energy}}$$

Joint computing and radio resource allocation problem

- Is there a task allocation in which all selfish MUs are satisfied?

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T \underbrace{(T_{i,a}^{c,off}(\mathbf{d}) + T_i^{c,exe}(\mathbf{d}))}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,a}^c(\mathbf{d})}_{\text{energy}}$$

Joint computing and radio resource allocation problem

- Is there a task allocation in which all selfish MUs are satisfied?
- Can it be computed using a decentralized algorithm?

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T \underbrace{(T_{i,a}^{c,off}(\mathbf{d}) + T_i^{c,exe}(\mathbf{d}))}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,a}^c(\mathbf{d})}_{\text{energy}}$$

Joint computing and radio resource allocation problem

- Is there a task allocation in which all selfish MUs are satisfied?
- Can it be computed using a decentralized algorithm?
- What is the complexity of the algorithm?

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,a}^c(\mathbf{d}) = \gamma_i^T \underbrace{(T_{i,a}^{c,off}(\mathbf{d}) + T_i^{c,exe}(\mathbf{d}))}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,a}^c(\mathbf{d})}_{\text{energy}}$$

Joint computing and radio resource allocation problem

- Is there a task allocation in which all selfish MUs are satisfied?
- Can it be computed using a decentralized algorithm?
- What is the complexity of the algorithm?
- How good are the system performance?

Computation Offloading Game

Interactions between MUs modeled as a strategic game $<\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$

Computation Offloading Game

Interactions between MUs modeled as a strategic game $<\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$

Best reply of a player

- Strategy that minimizes the cost of a player given the other players' strategies

Computation Offloading Game

Interactions between MUs modeled as a strategic game $<\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$

Best reply of a player

- Strategy that minimizes the cost of a player given the other players' strategies

Nash Equilibrium

- Strategy profile d^* in which every player's strategy is a best reply to the other players' strategies
  - $C_i(d_i^*, d_{-i}^*) \leq C_i(d_i, d_{-i}^*)\ \forall d_i \in \mathfrak{D}_i, \forall i \in \mathcal{N}$

Elastic cloud

Elastic cloud

NE exists in the case of an elastic cloud

Elastic cloud

NE exists in the case of an elastic cloud

Sketch of Proof

The computation offloading game with elastic cloud admits the generalized ordinal potential function

$$\Phi(\mathbf{d}) = \sum_{a=1}^{A} \sum_{n=1}^{n_a(\mathbf{d})} log(n) - \sum_{a=1}^{A} \sum_{i=1}^{N} log(M_i R_{i,a}) I(d_i, a),$$

$$M_i = \frac{\gamma_i^E v_i + \gamma_i^T (1/F_i^0 - 1/F^c)}{\gamma_i^T + \gamma_i^E P_i} \cdot \frac{L_i}{D_i}$$

Elastic cloud

NE exists in the case of an elastic cloud

Sketch of Proof

The computation offloading game with elastic cloud admits the generalized ordinal potential function

$$\Phi(\mathbf{d}) = \sum_{a=1}^{A} \sum_{n=1}^{n_a(\mathbf{d})} log(n) - \sum_{a=1}^{A} \sum_{i=1}^{N} log(M_i R_{i,a}) I(d_i, a),$$

$$M_i = \frac{\gamma_i^E v_i + \gamma_i^T (1/F_i^0 - 1/F^c)}{\gamma_i^T + \gamma_i^E P_i} \cdot \frac{L_i}{D_i}$$

ImprovementPath algorithm

- Starts from an arbitrary initial strategy profile
- One MU at a time is allowed to perform an improvement step

Elastic cloud

NE exists in the case of an elastic cloud

Sketch of Proof

The computation offloading game with elastic cloud admits the generalized ordinal potential function

$$\Phi(\mathbf{d}) = \sum_{a=1}^{A} \sum_{n=1}^{n_a(\mathbf{d})} log(n) - \sum_{a=1}^{A} \sum_{i=1}^{N} log(M_i R_{i,a}) I(d_i, a),$$

$$M_i = \frac{\gamma_i^E v_i + \gamma_i^T (1/F_i^0 - 1/F^c)}{\gamma_i^T + \gamma_i^E P_i} \cdot \frac{L_i}{D_i}$$

ImprovementPath algorithm

- Starts from an arbitrary initial strategy profile
- One MU at a time is allowed to perform an improvement step

Result

- ImprovementPath algorithm terminates in a NE after a finite number of steps in the case of an elastic cloud

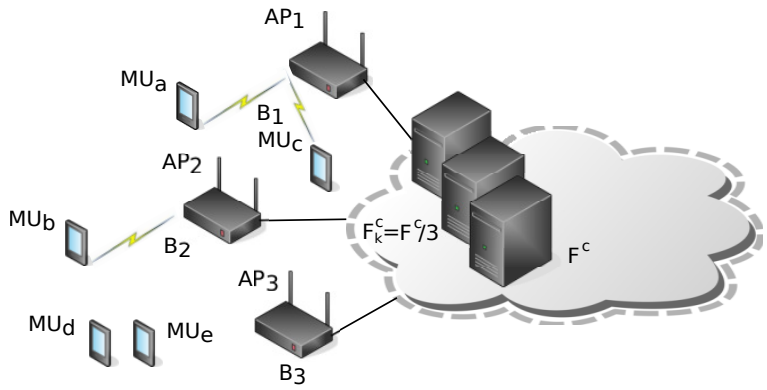
Non-elastic cloud

Non-elastic cloud

ImprovementPath algorithm may cycle in the case of a non-elastic cloud

Non-elastic cloud

ImprovementPath algorithm may cycle in the case of a non-elastic cloud



$$(1,2,1,0,0) \underset{c}{\rightarrow} (1,2,2,0,0) \underset{b}{\rightarrow} (1,0,2,0,0) \underset{d}{\rightarrow} (1,0,2,2,0) \underset{e}{\rightarrow} (1,0,2,2,2)$$
$$\underset{c}{\rightarrow} (1,0,1,2,2) \underset{b}{\rightarrow} (1,3,1,2,2) \underset{e}{\rightarrow} (1,3,1,2,0) \underset{d}{\rightarrow} (1,3,1,0,0) \underset{b}{\rightarrow}$$
$$(1,2,1,0,0)$$

Non-elastic cloud

Non-elastic cloud

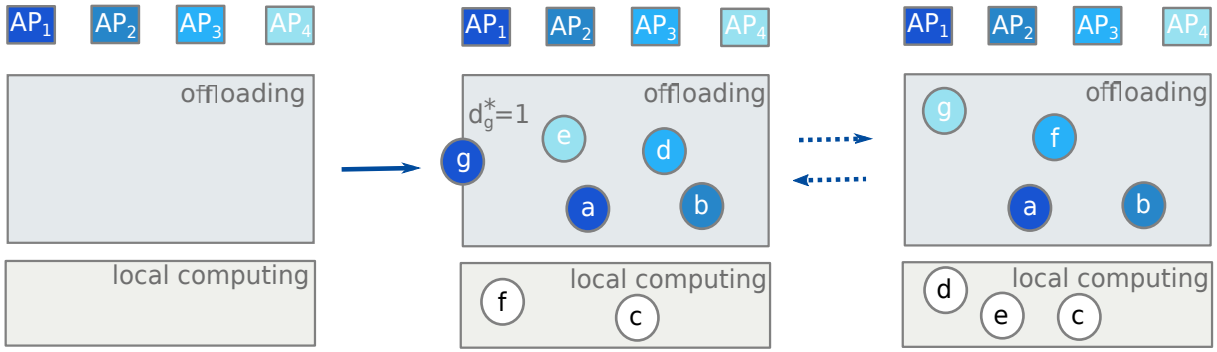
NE exists in the case of a non-elastic cloud

Non-elastic cloud

NE exists in the case of a non-elastic cloud

Join and Play Best Reply (JP-BR) algorithm

- Starts from an empty system
- MUs join a system one at a time and play a best reply
- MUs can perform best improvements one at a time

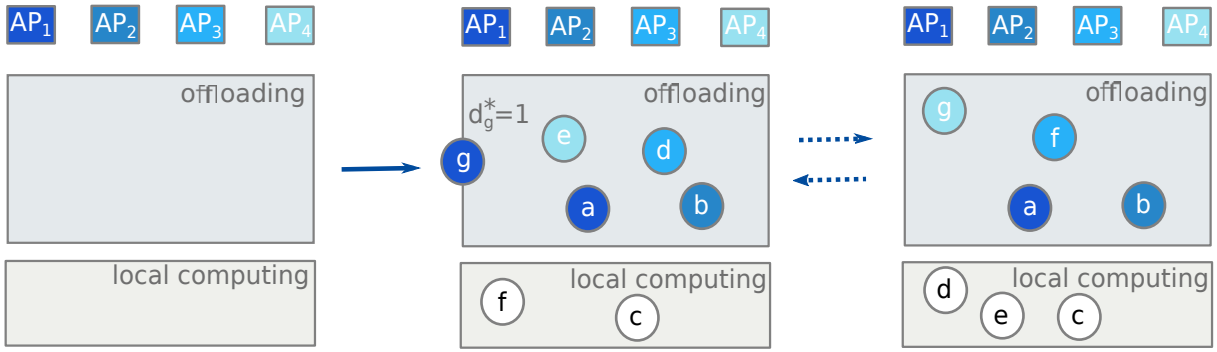


Non-elastic cloud

NE exists in the case of a non-elastic cloud

Join and Play Best Reply (JP-BR) algorithm

- Starts from an empty system
- MUs join a system one at a time and play a best reply
- MUs can perform best improvements one at a time



Results

- JP-BR algorithm terminates in a NE both in the case of elastic and in the case of non-elastic cloud
- Computational complexity of the JP-BR algorithm $\sim O(AN^2)$

Price of Anarchy (PoA)

- Ratio of the worst case NE cost and the minimal cost

$$PoA = \frac{\max\limits_{\mathbf{d}^*} \sum_{i \in \mathcal{N}} C_i(\mathbf{d}^*)}{\min\limits_{\mathbf{d} \in \mathfrak{D}} \sum_{i \in \mathcal{N}} C_i(\mathbf{d})}$$

Price of Anarchy (PoA)

- Ratio of the worst case NE cost and the minimal cost

$$PoA = \frac{\max\limits_{\mathbf{d}^*} \sum_{i \in \mathcal{N}} C_i(\mathbf{d}^*)}{\min\limits_{\mathbf{d} \in \mathfrak{D}} \sum_{i \in \mathcal{N}} C_i(\mathbf{d})}$$

- Upper bound on the PoA for the computation offloading game:

$$PoA \leq \frac{\sum_{i \in \mathcal{N}} C_i^0}{\sum_{i \in \mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}\}}$$

  - $C_{i,a}^{\bar{c}}$ is the offloading cost of MU i via AP a when she offloads alone

Equilibrium Solution vs. Optimal Solution

Evaluation scenario

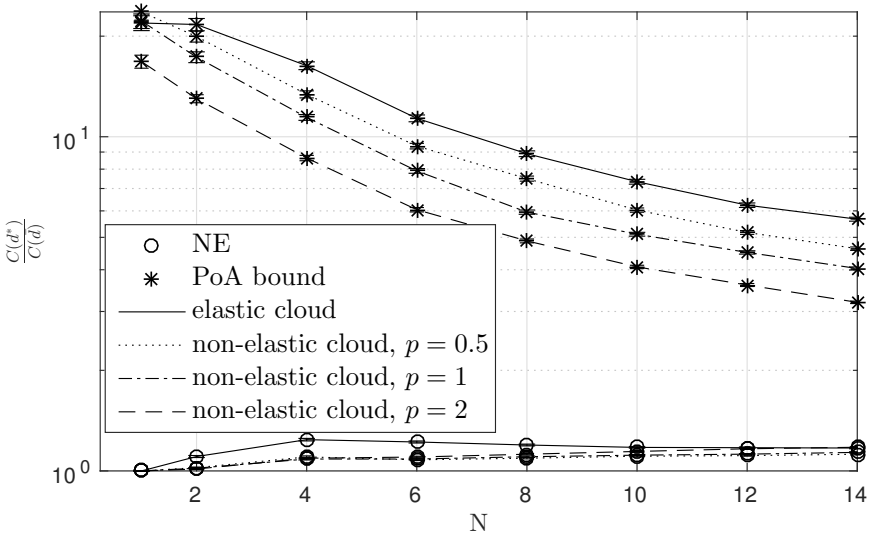
- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{pn(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

Equilibrium Solution vs. Optimal Solution

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{pn(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

Equilibrium cost vs. optimal cost (3 APs)



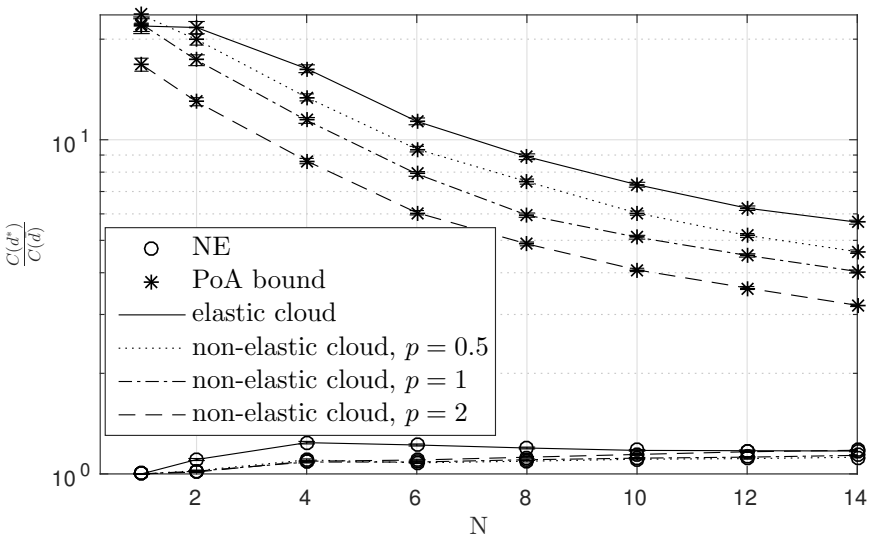
- Equilibrium cost is close to the optimal cost

Equilibrium Solution vs. Optimal Solution

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{pn(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

Equilibrium cost vs. optimal cost (3 APs)



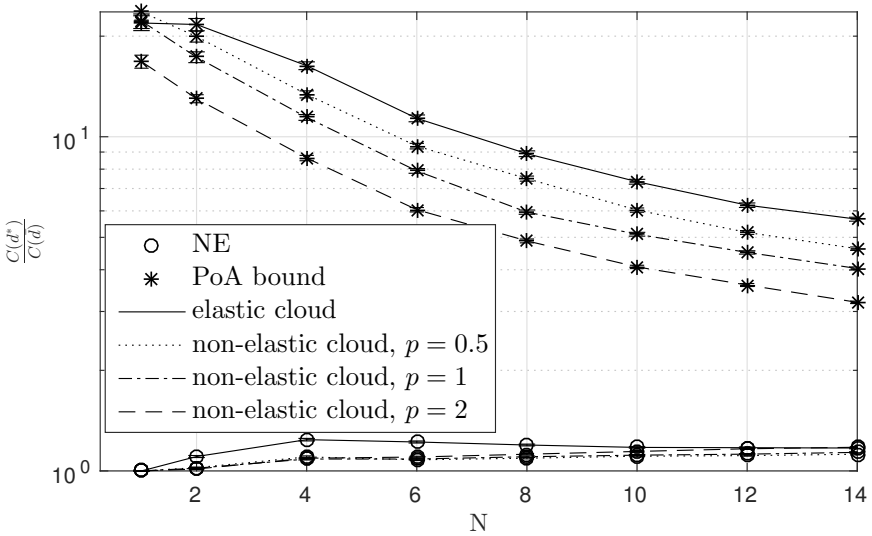
- Equilibrium cost is close to the optimal cost
- Cost ratio is fairly insensitive to the number of MUs

Equilibrium Solution vs. Optimal Solution

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{pn(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

Equilibrium cost vs. optimal cost (3 APs)



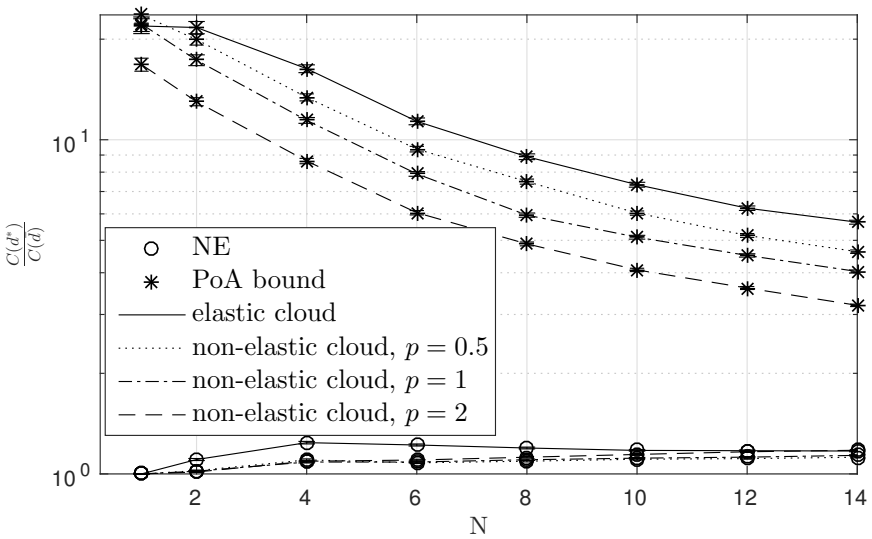
- Equilibrium cost is close to the optimal cost
- Cost ratio is fairly insensitive to the number of MUs
- Upper bound on the PoA decreases as the cloud provisioning coefficient p increases and as the number of MUs increases

Equilibrium Solution vs. Optimal Solution

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{pn(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

Equilibrium cost vs. optimal cost (3 APs)



- Equilibrium cost is close to the optimal cost
- Cost ratio is fairly insensitive to the number of MUs
- Upper bound on the PoA decreases as the cloud provisioning coefficient p increases and as the number of MUs increases

Impact of the input data size

Evaluation scenario

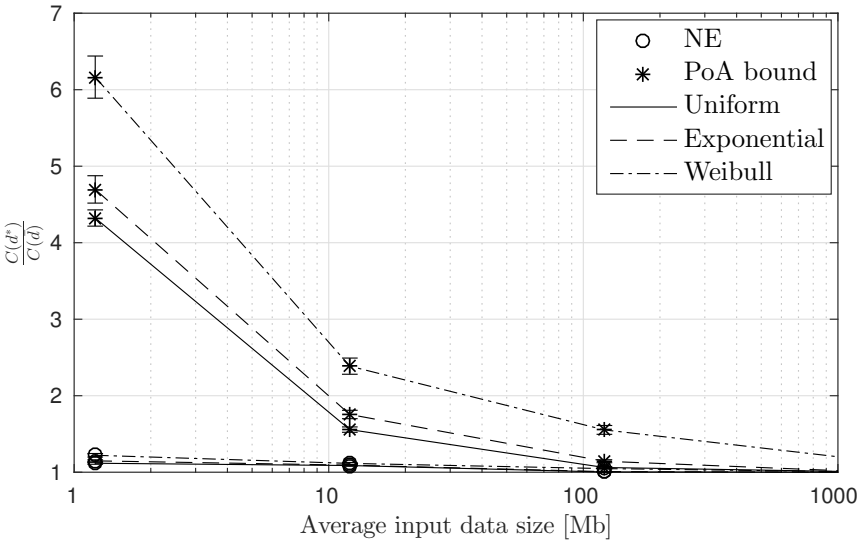
- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim$ uniform, exponential and Weibull distributions
- Complexity of the task $L_i = 0.45$ *Gigacycles*

Impact of the input data size

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim$ uniform, exponential and Weibull distributions
- Complexity of the task $L_i = 0.45$ *Gigacycles*

Equilibrium cost vs. optimal cost (3 APs and 12 MUs)



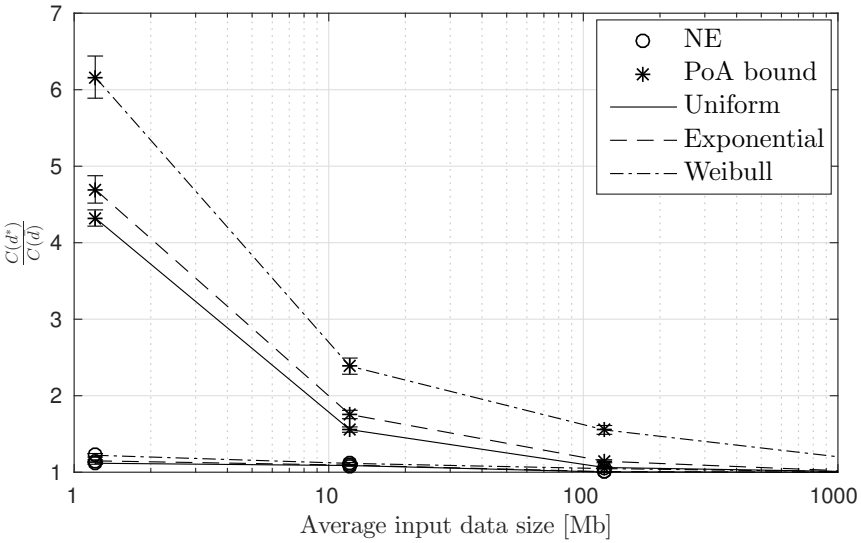
- Cost ratio is fairly insensitive to the mean input data size

Impact of the input data size

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim$ uniform, exponential and Weibull distributions
- Complexity of the task $L_i = 0.45$ *Gigacycles*

Equilibrium cost vs. optimal cost (3 APs and 12 MUs)



- Cost ratio is fairly insensitive to the mean input data size
- Upper bound on the PoA decreases with the mean input data size and for large data sizes it reaches the cost ratio

Computational complexity

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

Computational complexity

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

JPBR algorithm - two orderings of adding MUs (10 APs and 100 APs)

- Random order
- LRF order - MUs are added in increasing order of their ratio $D_i/(C_i^0 L_i)$

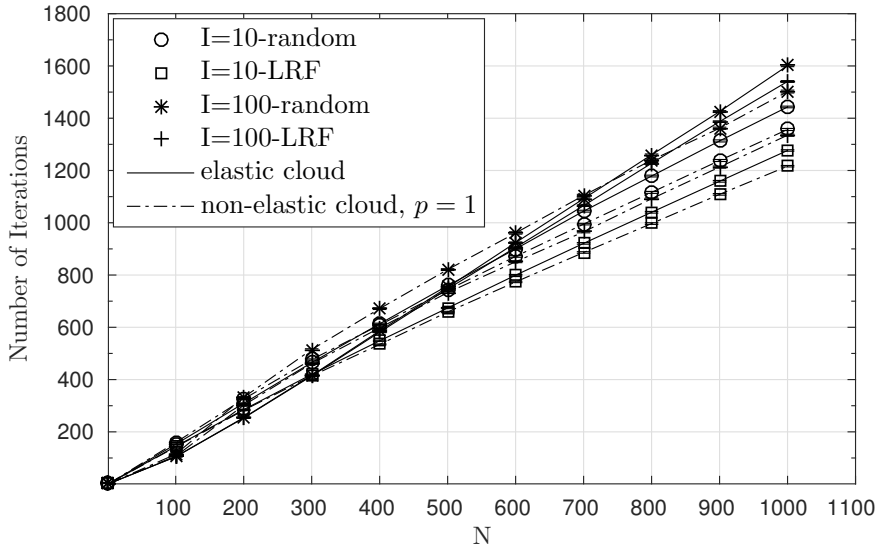
Computational complexity

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

JPBR algorithm - two orderings of adding MUs (10 APs and 100 APs)

- Random order
- LRF order - MUs are added in increasing order of their ratio $D_i/(C_i^0 L_i)$



- Number of iterations is fairly insensitive to the order of adding the MUs

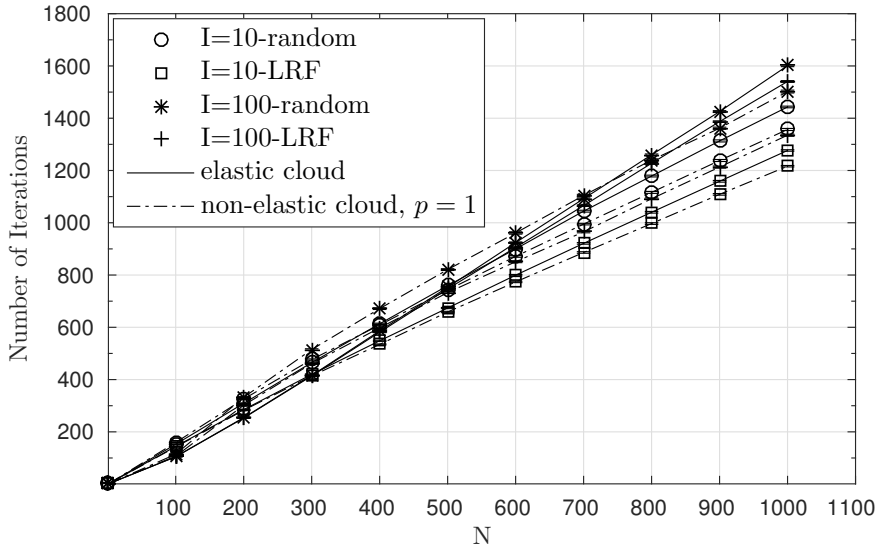
Computational complexity

Evaluation scenario

- MUs are placed at random over a square area of $1km \times 1km$
- $F^c = 100$ *Gcycles*, non-elastic cloud: $F_i^c(n(\mathbf{d})) = \frac{F^c}{n(\mathbf{d})}$, $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Input data size of the task $\sim \mathcal{U}(0.42, 2)$ Mb
- Complexity of the task $\sim \mathcal{U}(0.1, 0.8)$ *Gigacycles*

JPBR algorithm - two orderings of adding MUs (10 APs and 100 APs)

- Random order
- LRF order - MUs are added in increasing order of their ratio $D_i/(C_i^0 L_i)$



- Number of iterations is fairly insensitive to the order of adding the MUs
- Number of iterations mostly depends on the number of MUs

Summary and Future Work

- Proposed decentralized algorithms for computing an efficient task allocation among selfish MUs
  - elastic cloud
  - non-elastic cloud

Summary and Future Work

- Proposed decentralized algorithms for computing an efficient task allocation among selfish MUs
  - elastic cloud
  - non-elastic cloud

- Interesting extensions
  - D2D collaborative computation offloading
  - unknown number of MUs that offload
  - stochastic model of task arrivals

A Game Theoretic Analysis of Selfish Mobile Computation Offloading

Slađana Jošilo, György Dán

Department of Network and Systems Engineering
School of Electrical Engineering, KTH Royal Institute of Technology

Atlanta, May 4, 2017